Spark Over a Mesos Secured Productive Environment
Index

1. Presentation
2. Use Case
3. Spark, Kerberos and Mesos
4. Dynamic authentication
5. Mutual TLS
6. Postgres
7. Network Isolation
8. Live Demo
9. Q&A
1 Presentation
After working with traditional processing methods, I started to do some R&S Big Data projects and I fell in love with the Big Data world. Currently I’m doing some awesome Big Data projects and tools at Stratio.
Coming from the Dark Side I found a place to grow and innovate at Stratio.

I am passionate about astrophysics, a compulsive SciFi consumer and I really enjoy automating and Dockerizing everything around me!
Our Company

Stratio accompanies businesses on their journey through complete Digital Transformation. Through hard work and creativity, our ambition is to reinvent companies around their data so they can compete in a changed world.

Our solution is a transformational product that uses third generation Big Data technologies to execute the most comprehensive form of Digital Transformation, ensuring scalability, maximum flexibility and adaptation to new markets.

If you have any questions or would like a demonstration of our product, get in touch: contact@stratio.com
Our product
2 Use Case
DC/OS

- DC/OS is an open-source distributed operating system, based on Apache Mesos kernel. DC/OS manages multiple services and its instances (whether on cloud or on-premise) from a single interface.

- Users can deploy containers for distributed services and legacy applications.

- DC/OS provides a network layer, service discovery and resource management.
DC/OS Architecture Layers

Software
- Services
- Custom Schedulers & Tasks
- Jobs

Platform
- Networking
- Security
- Container Orchestration
- Cluster Management
- Logging & Metrics
- Storage

Infrastructure
- Cloud
- Bare Metal

Additional Components:
- Package Repository
- Browser GUI
- Terminal CLI
- Container Repository
DC/OS Use Case: preconditions

➢ Clients look for user profiling and highly scalable distributed services.

➢ The access to these resources must be done using secured connections.

➢ Not all services are secured using the same protocol.

➢ Clients don’t want to manage their secrets.

➢ Spark is Stratio’s distributed processing framework.
Spark <3 Mesos

➢ Spark was born as a Mesos use case validation.

➢ Spark can be deployed both in a client and cluster mode over mesos.

➢ In enterprise environments only the cluster mode is recommended.
Main {
    ....
    ....
    sc = new SparkContext()
    rdd1 = sc.textFile.map()
    rdd1.saveAs.... 
}

Main {
    ....
    ....
    sc = new SparkContext()
    rdd1 = sc.textFile.map()
    rdd1.saveAs.... 
}

Spark dispatcher

Mesos Master

HDFS
Adding Kerberos to the equation

➢ HDFS was the first datastore certificated by Stratio.

➢ Kerberos is used as Hadoop security protocol (HDFS and YARN).

➢ It handles users and systems identities.

➢ It is based on principal, keytab and delegated tokens:
  - **Principal**: user or service identity
  - **Keytab**: file that contains a key associated to the principal
  - **Tokens**: Hadoop uses delegated Tokens
Adding Kerberos to the equation

- Spark using Mesos did not allow any Kerberos options.

- Stratio Spark team integrates Spark’s Kerberos functionalities (1.6.2, 2.1.0 and 2.2.0).

- New functionality added: using proxy user for each task.

- All is explained in Spark Summit East 2017.
read.format(parquet).path("/user1/b").option("user", "user2")
toDF.saveAsParquet("b")
textFile("a")
saveAsTextFile("a")

Driver

Executor

HDFS

Kerberos

sparkUser
user1
user2

Stratio’s Solution
4 Dynamic authentication
➢ Each technology will have its own security protocol.

➢ The secrets have to be accessible from all the machines inside the cluster.

➢ Stratio security team integrated a KMS in the Stratio platform to store and manage all the platform secrets.
Dynamic authentication

➢ Ok, we have a secret storage. But how do our processes access it in a secured way?

➢ Who keeps the key that opens the
Spark - KMS

DCS cluster

- Stratio Spark Dispatcher
- Mesos Master
- Mesos Agent-1
- Marathon
- Mesos Agent-2
- Spark Driver
- Mesos Agent-3
- Mesos Agent-4
- Spark Application
- AppToken

HDFS

Vault
The application token is written in executor logs.

Sensitive information is sent using a non-secured transportation layer.

The application log is also written in...
Spark - KMS Dynamic Authentication

DCS cluster

Stratio Spark Dispatcher

Mesos Master

Mesos Agent-1

Marathon

Mesos Agent-2

Spark Driver

Stratio Spark Dispatcher

Spark Executor

Mesos Agent-3

secretID/roleID

Mesos Agent-4

HDFS

Vault

SecretID/roleID

role

appToken

SecretID/roleID
Spark - KMS Dynamic Authentication

★ Non sensitive information is shared.

★ If there is any token intervention we can detect it.

★ No sensible information is stored in...

👋 Our secrets are safe!
Long handshakes are always bad!!
Mutual TLS Datastores

Mutual authentication or two-way authentication refers to two parties authenticating each other at the same time, being a default mode of authentication in some protocols.

Most Big Data technologies allows the implementation of Mutual TLS, which provides a way to identify clients, servers and multiple instances for services.
Mutual TLS Datastores

Superman

Batman

Keystore

Truststore

Keystore

Truststore
Mutual TLS Datastores

Wikipedia:

“[…] As it requires provisioning of the certificates to the clients and involves less user-friendly experience, it's rarely used in end-user applications.”
Mutual TLS Datastores

Stratiopedia:

“By default most users are not used to playing with big data things. Let’s do things as easy as we can, so hack something with that TLS thing”
Mutual TLS Datastores

```scala
def main(args: Array[String]): Unit = {
  val sparkConf = new SparkConf().setAppName("ElasticSearch_ATJob")
  val elasticOptions = extractElasticSecurityOption(args(0), args(1), sparkConf)
  val spark = SparkSession.builder().config(sparkConf).getOrCreate()
  spark.sparkContext.setLogLevel("DEBUG")

  elasticOptions.foreach({
    case (key, value) => println(s"KEY: $key, VALUE: $value")
  })

  val dataset = spark
    .read
    .format("org.elasticsearch.spark.sql")
    .options(elasticOptions)
    .load()
  println(dataset.rdd.getNumPartitions)
  dataset.printSchema()
  dataset.show(10)
}

  val options = Map( "es.index.auto.create" -> "true", "es.nodes" -> host, "es.resource" -> "test/type1", "es.port" -> port )
  val prefixSparkElastic = "spark.ssl.datastore."
  val prefixElasticSecurity = "es.net.ssl"

  if (sparkConf.getOption(prefixSparkElastic + "enabled").isDefined && sparkConf.get(prefixSparkElastic + "enabled") == "true") {
    val configElastic = sparkConf.getAllWithPrefix(prefixSparkElastic).toMap
    options ++ Map( s"${prefixElasticSecurity}.keystore.pass" -> configElastic("keyStorePassword"), s"${prefixElasticSecurity}.keystore.location" -> s"file:${configElastic("keyStore")}" )
    options ++ Map( s"${prefixElasticSecurity}.truststore.pass" -> configElastic("trustStorePassword"))
  } else {
    Map()
  }
```

Spark - Mutual TLS Datastores
6 Postgres
Postgres

➢ Just when everything seemed to work harmoniously a new use case appeared needing to access a secure postgres.

➢ JDBC connection chain includes a **PKCS8** format certificate.

➢ No parsing method exists from pem to PKCS8.
Postgres

- **First approach:** spark-env.sh + openssl hack.

- **Nowadays:** Scala class to parse from pem to pkcs8.

- **Future Improvements:** Provide a SSLSocketFactory.
7 Network isolation
Are you sure that your cluster is secure?
Stratio security team integrates Mesos with SDN (Software Defined Networks).

SDN allows us to create isolated virtual networks with access to the resources defined for each user.

SDN does not add any limitation to the Mesos architecture.

We can assign just the resources needed by each user.

Mesos still has its awesome elasticity.
Are you sure that your cluster is secure?
7 Live Demo
WE ARE HIRING
people@stratio.com
@StratioBD